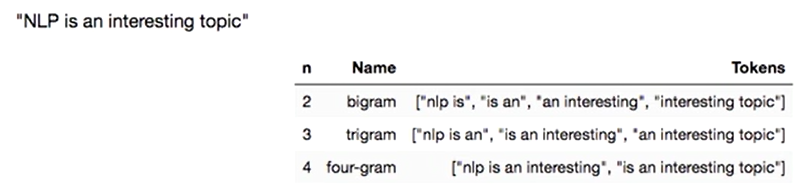
*Start a new Python project folder … do not reuse the previous workspace … As with any keyboard-driven console-like environment, developing muscle -memory for the common commands is also part of the learning curve.*

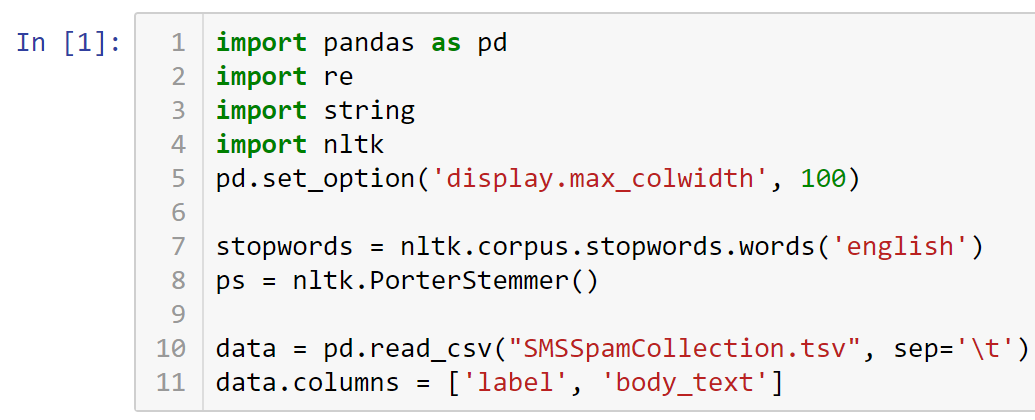
Vectorizing Raw Data: N-Grams

N-Grams creates a document-term matrix where counts still occupy the cell but instead of the columns representing single terms, they represent all combinations of adjacent words of length n in your text.



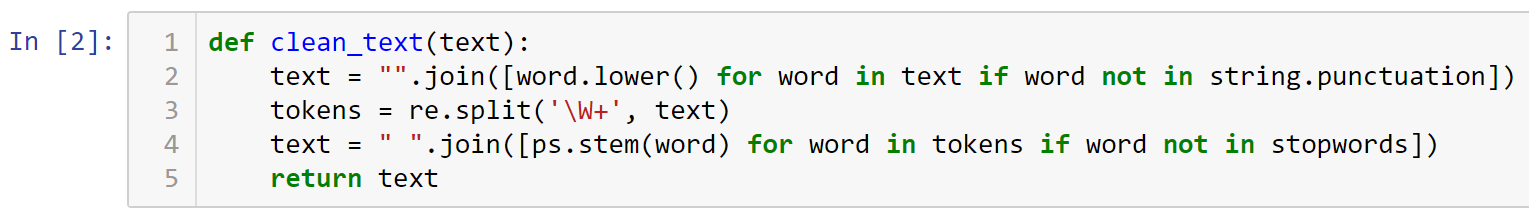
In the table above, if n=2, then it is called a bigram; if n=3 then it is called a trigram, etc. You can make n as small and as large as you want. There is usually an optimal value of n that will yield the best performance. So generally, you will tune this value to figure out which generates the best model. Google’s auto-complete uses N-grams approach, as an example. If you type “natural language,” Goggle knows that you are most likely looking for “natural language processing,” so it might suggest that as a full phrase that you’d like to search for.

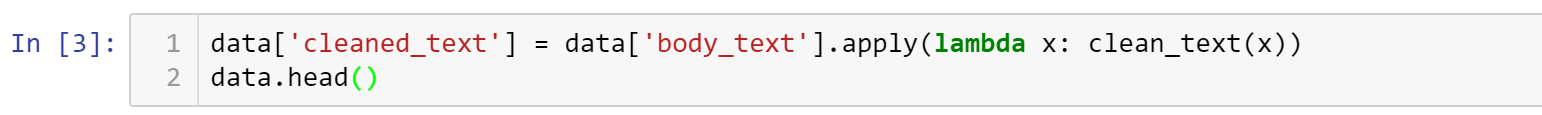
Let’s jump to the code:



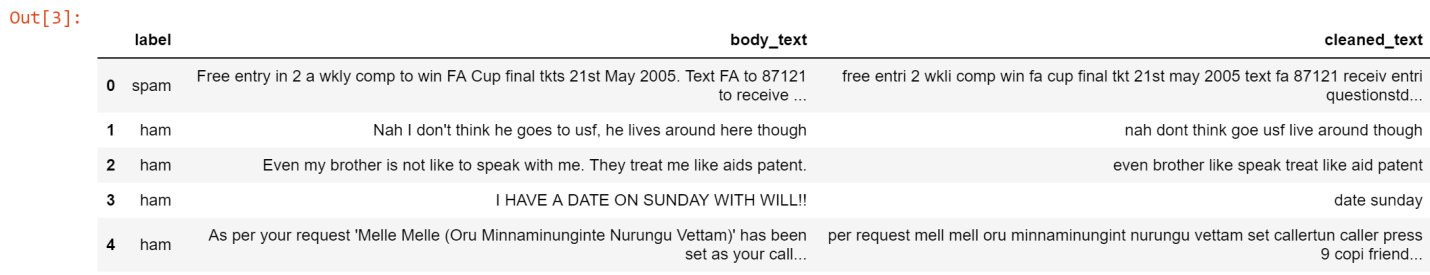
Create function to remove punctuation, tokenize, remove stopwords, and stem …

The cleaning side is slightly different from what we did previously. Note the blank space before .join in line #4.



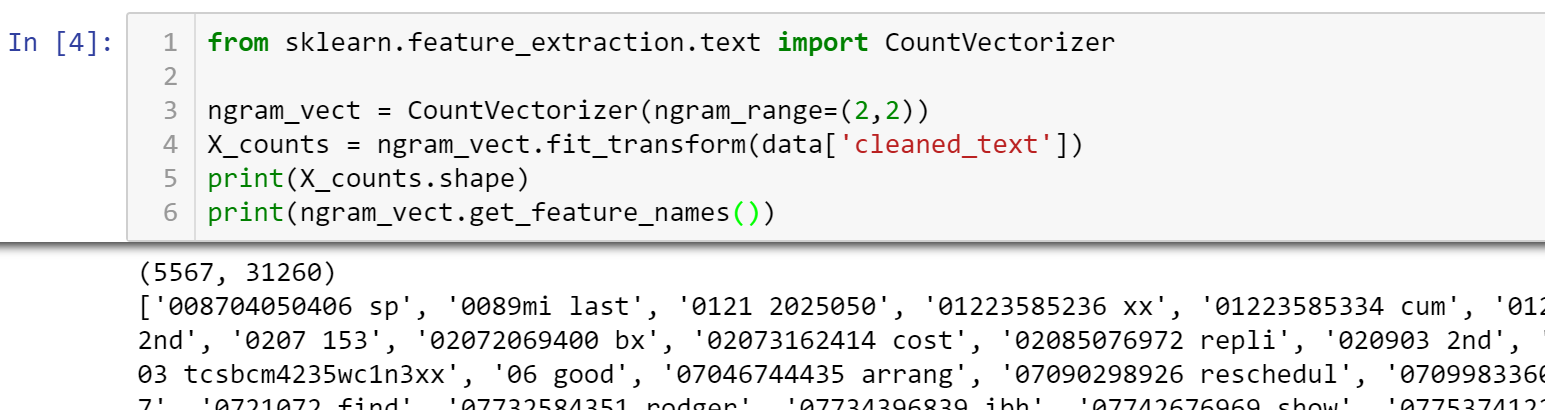


You can see in the output below that the tokenized list is constructed back to a sentence.



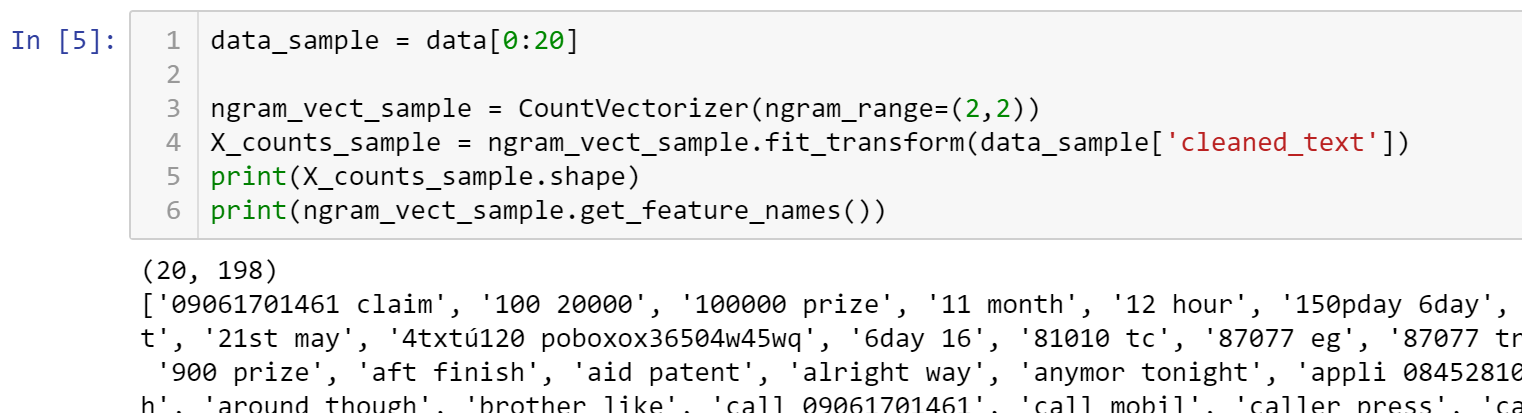
Applying CountVectorizer (w/ N-Grams)

The parameters in CountVectorizer that declare the ngram\_range to look means 1=unigram; 2=bigram; 3=trigram; etc. (2, 2) means we are only looking for bigrams.

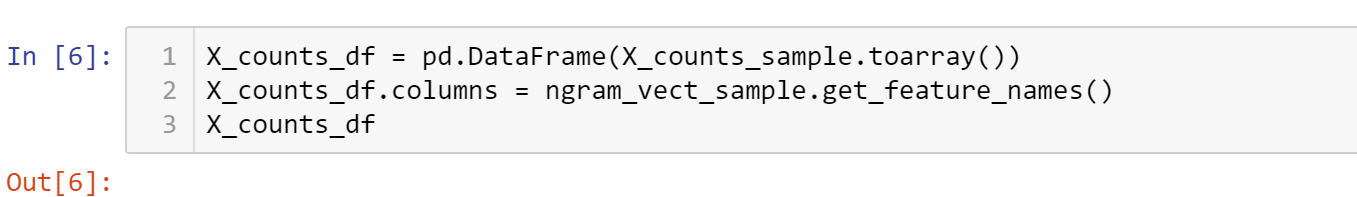


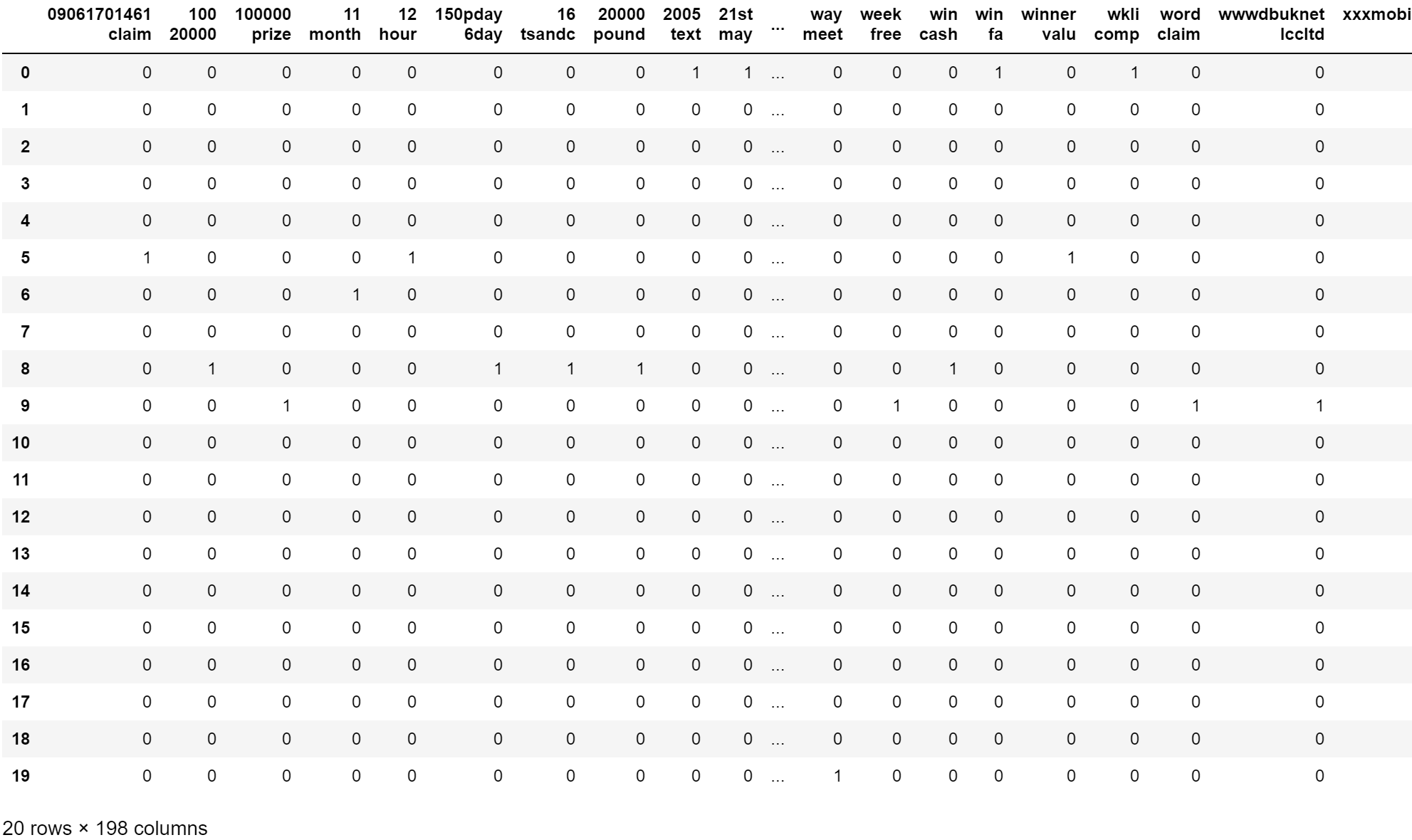
We still have the 5,567 rows, but the columns are now reduced to 31,260. That means 31,260 unique combinations of 2 words. Go down the output and note the two-words unique combinations.

Let’s jump to a smaller sample so we can visualize the data better. So applying CountVectorizer (w/ N-Grams) to smaller sample …



Now, instead of 31,260 columns, we have 198. Just like in the previous exercise, we know that the row output is a sparse matrix. This is a matrix in which most entries are 0. In the interest of efficient storage, a sparse matrix will be stored by only storing the locations of the non-zero elements. So we will convert this to a DataFrame and add the proper column labels.





For example, the output will tell you that “2005 text” and “21st may” occurred in the very first text message, but not in any of the other text messages. The value of N-Grams is that it provides more context to words. Increasing N-Grams from bigrams (i.e. 2) to 7-Grams allowed you to create more meaningful unique context.

* All submissions should be separate from other exercises and quests. Please do not lump all your answers into one document and re-using that same workspace to gain multiple points. Thanks.
* Place your name at the bottom of your code, download your Python program in html format, and submit your work in Canvas.